CORBA and HLA: Enabling Future Network-Centric Vehicle Systems?

William Pritchett & William Protzman
DCS Corporation

Robert Kling
US Army TACOM

Abstract

Increased emphasis on simulation-based acquisition and network-centric warfare within the objective force has posed key challenges within the embedded ground vehicle community. While simulation-based acquisition enables a faster time to field through increased concurrency, tighter decision cycles, more efficient and effective testing; the ultimate goal of rapidly transitioning outputs from the simulation phase of a program to the vehicle design and integration phase is minimally met in today's environment due to varied architectures, levels of fidelity, and design methodologies. In the same respect, the dynamic nature of network-centric warfare further strains legacy architectures, as they cannot handle the new and often dynamic operational requirements. Two technologies currently under consideration with respect to their ability to solve some or all of these design and architectural gaps are the Common Object Request Broker Architecture (CORBA) family of services and the High-Level Architecture (HLA). CORBA is a distributed object architecture developed by the Object Management Group (OMG), and though originally developed for information systems, has been extended to accommodate real-time and embedded requirements. HLA is the Department of Defense mandated architecture for distributed simulation. HLA includes a runtime infrastructure (RTI) and one or more federates combined to form the simulation system, or federation. Though both CORBA and HLA are used to build complex, distributed systems, there are differences in these two architectures that complicate their use within a single system. This paper evaluates the suitability of both CORBA and HLA for use in ground combat vehicles, as both stand-alone technologies and in conjunction with one another.

1 Background

To better respond to changing world threats, the Army is in the midst of a major force transformation grounded in concepts for future joint and combined operations. At the center of the transformation is the unit of action, a strategically responsive force, rapidly deployable anywhere in the world in 96 hours after liftoff, providing overmatching lethality with advanced survivability against any threat, versatile and responsive to the needs of the Joint Task Force Commander, while able to transition rapidly between missions, tactical engagements and battles with minimal organizational adjustment. The makeup of a unit of action includes a highly-networked combination of manned and unmanned ground vehicles being developed under the auspices of the Future Combat System (FCS).

From the ground combat vehicle weapon system perspective, the paradigm shift catalyzed by the FCS program is based in an accelerated convergence of architectures (depicted in Figure 1). These architectures are the Platform Architecture (e.g., vehicle computers, weapons, mobility, crew stations, and electronics), C4ISR Architecture (e.g., sensors, battlefield command and control, situational awareness), and Modeling and Simulation Architecture (e.g., computer-based and embedded training, mission rehearsal, terrain registration).

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
14 APR 2003	**Journal Article**	**06-02-2003 to 16-03-2003**

4. TITLE AND SUBTITLE
CORBA and HLA: Enabling Future Network-Centric Vehicle Systems?

5a. CONTRACT NUMBER
5b. GRANT NUMBER
5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
William Pritchett; William Protzman; Robert Kling

5d. PROJECT NUMBER
5e. TASK NUMBER
5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
DCS Corporation,7400 Miller Dr,Warren,Mi,48397-5000

8. PERFORMING ORGANIZATION REPORT NUMBER
; #13861

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. Army TARDEC, 6501 East Eleven Mile Rd, Warren, Mi, 48397-5000

10. SPONSOR/MONITOR'S ACRONYM(S)
TARDEC

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
#13861

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Increased emphasis on simulation-based acquisition and network-centric warfare within the objective force has posed key challenges within the embedded ground vehicle community. While simulation-based acquisition enables a faster time to field through increased concurrency, tighter decision cycles, more efficient and effective testing; the ultimate goal of rapidly transitioning outputs from the simulation phase of a program to the vehicle design and integration phase is minimally met in today?s environment due to varied architectures, levels of fidelity, and design methodologies. In the same respect, the dynamic nature of network-centric warfare further strains legacy architectures, as they cannot handle the new and often dynamic operational requirements. Two technologies currently under consideration with respect to their ability to solve some or all of these design and architectural gaps are the Common Object Request Broker Architecture (CORBA) family of services and the High-Level Architecture (HLA). CORBA is a distributed object architecture developed by the Object Management Group (OMG), and though originally developed for information systems, has been extended to accommodate real-time and embedded requirements. HLA is the Department of Defense mandated architecture for distributed simulation. HLA includes a runtime infrastructure (RTI) and one or more federates combined to form the simulation system, or federation. Though both CORBA and HLA are used to build complex, distributed systems, there are differences in these two architectures that complicate their use within a single system. This paper evaluates the suitability of both CORBA and HLA for use in ground combat vehicles, as both stand-alone technologies and in conjunction with one another.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**Public Release**	**11**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

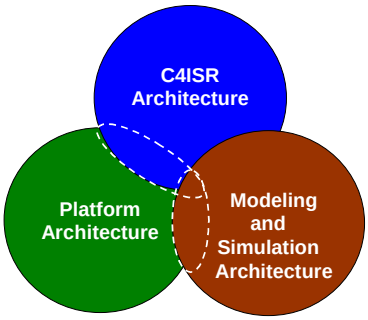

Figure 1 Platform, C4ISR, and Simulation Architecture Relationships

There is, however, an underlying barrier present in today's environment that limits the ability to leverage across these architectures. That barrier is that these architectures are grounded in distinct domains, which have traditionally focused on and evolved from discrete technologies and infrastructure to realize the development and integration of resultant domain applications. For example, platform applications tend to drive towards embedded real-time systems, imposing resource and timing constraints; C4ISR applications tend to be developed as enterprise/information technology systems targeted for workstation class computers; and modeling and simulation applications tend to focus on force simulations, system-level model based acquisition, and training.

Today, cross-architecture technologies are typically incorporated into Army vehicle weapon systems via a bolt on vs. integrated approach. An example is an FBCB2 appliqué computer running a command and control/situational awareness application providing its own display and integrated to the vehicle system via Ethernet and radio interfaces. As specific FCS vehicle system(s) architectures are designed and implemented, a high degree of cohesion among these three converging architectures will be not only desired, but necessary in order to integrate/leverage required technologies in accordance with the Objective Force system/system of system concepts and requirements (to include acquisition, development, deployment, management and sustainment).

An approach to facilitating this cross-architecture application reuse/leveraging within FCS can be realized through the analysis of the application infrastructure (distribution middleware) and identification of an approach to better correlate these middleware layers leading to application insertion as opposed to application and computer bolt on integration.

Figure 2 Platform, C4ISR, and Simulation Domain Middleware

Figure 2 depicts the three architectures, highlighting the middleware dominant within the application domains. Applications within the Platform Domain utilize a real-time operating system (RTOS) and often incorporate an operating environment (OE) middleware isolation layer, isolating hardware dependencies and maximizing application portability. Applications within the C4ISR Domain utilize workstation-based operating systems (e.g., Unix, Linux) and incorporate CORBA services. Applications within the M&S domain utilize a variety of operating systems (dependent on the simulation) and are developed in HLA and integrated via the HLA RunTime Infrastructure (RTI).

This paper provides an analysis of HLA and CORBA to identify their suitability to facilitate the leveraging, development, and integration of co-resident C4ISR and M&S applications within the ground vehicle weapon system platform.

1.1 High-Level Architecture

The High-Level Architecture (HLA) [1], shown in Figure 3, is a component-based software architecture developed by the U.S. Department of Defense during the 1990's to provide a low-cost, high-capability simulation infrastructure. The architecture calls for a *federation* of individual simulations called *federates*. A federate can represent varying levels of abstraction from a cockpit simulator to an entire fighting force. A *federation execution* is a session of a federation executing together. Along with individual federates, a federation contains a runtime infrastructure (RTI) and a common object model for data exchanged between federates in a federation, call the Federation Object Model (FOM). The specification of the architecture contains three main components: 1) ten rules that define relationships among federation components, 2) an object model template which specifies the form in which simulation elements are described, and 3) an interface specification that describes the way simulations interact during execution.

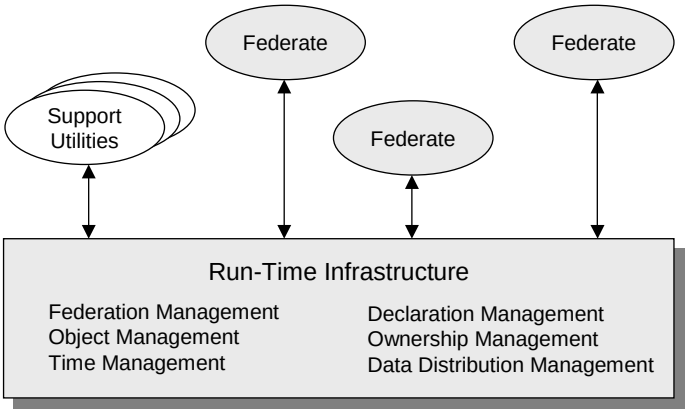

Figure 3 High Level Architecture

Of the ten rules specified by the architecture, five are federation rules and five are federate rules. The federation rules are as follows: 1) the federation shall have a FOM and shall be in object model template (OMT) format; 2) all representation of objects shall be in the federates and not the RTI; 3) during federation execution, all exchange of FOM data shall be via the RTI; 4) during federation execution, all federates shall interact with the RTI in accordance with the interface specification; and 5) during federation

execution, an attribute of an instance of an object may be owned by only one federate at a given time. The federate rules are: 1) federates shall have a SOM in OMT format; 2) federates shall be able to update/reflect attributes and send/receive data in accordance with their SOM; 3) federates shall be able to transfer/accept attribute ownership in accordance with their SOM; 4) federates shall be able to vary the conditions under which they provide attribute updates in accordance with their SOM; and 5) federates shall be able to manage the local time in a way which will allow them to coordinate data exchange with other members of the federation.

The object model template provides a common framework for HLA object model documentation and fosters interoperability and reuse of simulations via the specification of a common representational framework. The FOM is a description of all shared information (objects, attributes, interactions, and parameters) essential to a particular federation. The Simulation Object Model (SOM) describes objects, attributes, and interactions in a particular simulation that *can* be used externally in a federation. An attribute is the named portion of an object's state. An interaction is a change in the sending object state that may cause a state change in another (receiving) object. A parameter is the information associated with an interaction provided by the sending object to the receiving object. Federates update attributes by providing the new instance attribute value for an attribute, and reflect attribute changes by receiving the new instance attribute value for an attribute.

The HLA interface specification defines the access to RTI services. The interface specification is provided as an application programmer interface (API) in several forms including C++, Ada 95 and Java. The interface specification has six basic RTI service groups: federation management, declaration management, object management, ownership management, time management, and data distribution management.

1.2 Common Object Request Broker Architecture

The Common Object Request Broker Architecture (CORBA) [2] is an open, vendor-independent architecture and infrastructure that computer applications use to work together over networks. Using the standard Internet Inter-ORB protocol (IIOP), a CORBA-based program from any vendor, on almost any computer, operating system, programming language, and network, can interoperate with a CORBA-based program from the same or another vendor, on almost any other computer, operating system, programming language, and network. The Object Management Group (OMG), a consortium of vendors and other distributed computing stakeholders, produces and maintains the family of CORBA-related specifications.

CORBA applications are composed of *objects*, individual units of running software that combine functionality and data. For each object type, developers define an interface using the OMG Interface Definition Language (IDL). The interface is the syntax part of the contract that the server object offers to the clients that invoke it. Any client that wants to invoke an operation on the object *must* use this IDL interface to specify the operation it wants to perform, and to marshal the arguments that it sends. When the invocation reaches the target object, the object request broker (ORB) uses the *same*

interface definition to unmarshal the arguments so that the object can perform the requested operation with them. The ORB then uses the interface definition to marshal the results for their trip back, and to unmarshal them when they reach their destination.

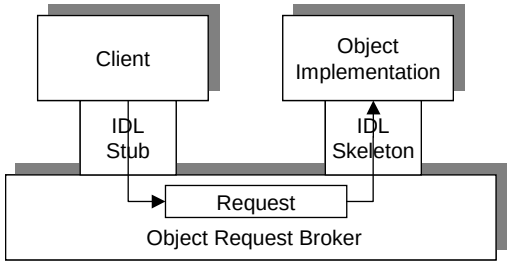

Figure 4 CORBA Architecture

This separation of interface from implementation, as shown in Figure 4, is the essence of CORBA—how it enables interoperability and location transparencies. Object interfaces have very strict definitions. In contrast, the ORB hides *implementation* details of an object - its running code, and its data from the rest of the system behind a boundary that the client may not cross. Clients access objects only through their advertised interface, invoking only those operations that the object exposes through its IDL interface, with only those parameters (input and output) that are included in the invocation.

Though originally developed for information systems, the OMG has created specialized CORBA extensions to address real-time and embedded systems. Most notable of the adopted specialized specifications are dynamic scheduling, minimum CORBA, CORBA event-service, CORBA notification service, and real-time CORBA. Real-time CORBA extends CORBA by facilitating the end-to-end predictability of activities in the system and by providing support for the management of resources. The dynamic scheduling specification extends real-time CORBA to encompass dynamically scheduled systems. The CORBA event and notification services provide services for asynchronous notification of events. Minimum CORBA defines a minimal subset of CORBA for resource constrained (embedded) environments.

The OMG also has related specifications under development including the data distribution service, real-time notification service, and reliable ordered multicast protocol. The data distribution service supports a Data-Centric Publish-Subscribe (DCPS) for real-time systems and offers, optionally, a Data Local Reconstruction Layer (DLRL). The real-time notification service will address priority ordering of events and deadline scheduling of events. Finally, the OMG is specifying a reliable ordered multicast protocol for those services, such as fault tolerance, that can benefit from such a protocol.

1.3 CORBA/HLA Relationship

Though dissimilar, there is a relationship between CORBA and the HLA. Developers can use CORBA IDL to specify the interface between federates and the RTI. Further, the OMG has specified the RTI itself in IDL and has standardized this interface as their Facility for Distributed Simulation Systems [3]. This means that CORBA-based

applications can instantiate RTI objects as CORBA objects and incorporate them into their system. Additionally, the nature of the ORB makes it a good candidate technology for implementing an HLA RTI, and in fact, ORB-based implementations of both RTIs and federates exist.

2 Suitability of HLA in Future Ground Vehicles

While HLA may be a suitable technology for large-scale distributed simulations, current implementations have major drawbacks as a solution for real-time embedded systems. Foremost is the lack of RTI implementations for real-time operating systems such as Wind River's VxWorks. There are, however, RTI products that run on Linux, an operating system gaining in popularity within the ground vehicle community. More investigation is needed, however, to determine whether or not Linux is a viable operating system for a total vehicle solution.

Another weakness to using HLA for ground vehicle embedded simulation is the fact that HLA does not specify timeliness criteria, thus limiting its usefulness in real-time simulations. McLean et. al. states the "lack of timeliness requirements, or any additional guidance, is a critical limitation for real-time simulation systems where the amount and predictability of RTI overhead is an important design factor" [4]. This is important because real-time distributed interactive simulation requires "a real-time response and predictable behavior from the end systems in order to interact with the physical world within the specific delay bounds and present data, images, audio, video, etc. to the users on a real-time basis" [5]. Numerous research efforts to extend HLA to real-time applications are addressing this problem [4][5].

The last major shortcoming to using HLA for real-time embedded simulation is the fact that HLA is not interoperable across languages or RTI implementations. Although HLA has multilanguage support, "the responsibility for interoperability between federates in different languages is placed on the RTI implementers" [6]. Further, HLA leaves the choice of protocol up to implementations. This forces systems into using a single RTI implementation and a single programming language. This is not always feasible within modern ground combat systems.

3 Suitability of CORBA in Future Ground Vehicles

While RTI has major limitations for use in embedded systems, CORBA is increasingly gaining acceptance in military domains as shown by the numerous programs that have fielded CORBA-based applications [7][8][9][10]. Common reasons cited for choosing CORBA include improved quality of service (QoS), predictable response times, small footprint, and real-time performance. One benchmarking study analyzed the performance of three available real-time ORBs and found data transfer rates a high as 5 MB/s using TCP/IP over a 10BaseT Ethernet on [11]. This performance is more than adequate for the closed loop control rates needed for most Vehicle Electronics (vetronics) applications. Further, CORBA's emphasis on interfaces over implementations leads to well-defined architectures that are more maintainable and supportable over the life cycle of a vehicle.

Additionally, the use of CORBA for middleware eliminates the need for application programmers to have detailed knowledge of the distribution mechanisms, thus allowing them to focus on the military-unique domain applications.

CORBA, however, is not without limitations too. The client/server paradigm used by CORBA may not be suitable for all ground vehicle applications. While this paradigm is proper for one-to-one object calls, network bandwidth may suffer in a data-oriented system in which data must flow from a single source to multiple destinations. The OMG is addressing this deficiency with their emerging data distribution specification. This specification will enable vehicle developers, where appropriate, to use the standards-based publish/subscribe paradigm in their software design to facilitate low-latency, high-bandwidth communications.

Another downside to using CORBA in vetronics applications is the lack of a suitable asynchronous notification mechanism. Though CORBA does have event and notification services, these do not allow for deadline-based event specification or event priorities, making it inefficient for real-time applications. The OMG is addressing these deficiencies with the emerging real-time notification service.

4 Heterogeneous HLA/CORBA-based System Example

This section provides an overview of an example system incorporating an HLA/CORBA based architecture to cohesively integrate a disparate M&S component into a ground vehicle weapon system platform. The example is taken from a current system design employed within the Army R&D community that interfaces an embedded training capability to a ground vehicle weapon system.

The current architecture employed within the embedded simulation system is presented in Figure 5.

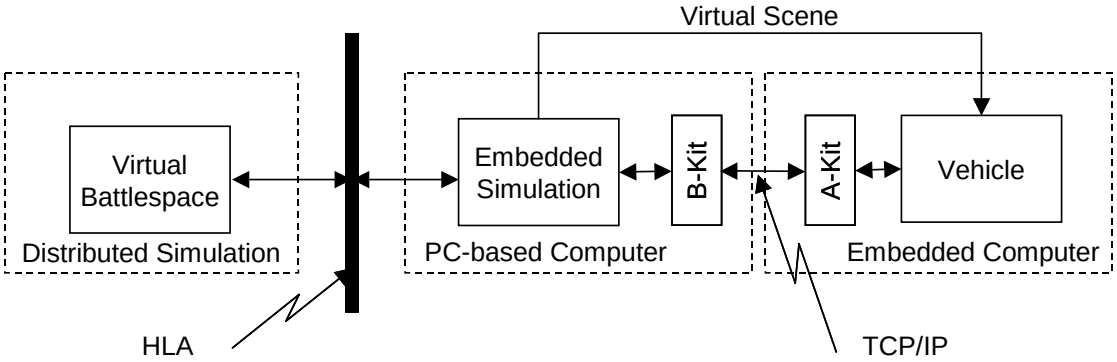

Figure 5 Embedded Simulation System

As depicted in the figure, a distributed simulation environment, representing force-level battlespace entities, is connected to an embedded simulation computer via HLA. The embedded simulation computer provides a gateway to the vehicle (the vehicle is represented in the figure as an embedded computer). The embedded simulation gateway enables the vehicle to participate in the force-level simulation as an individual entity by

providing correlated virtual world sensor views to the vehicle and representing the vehicle via operator and system interaction (e.g., mobility, targeting, firing, …) within the virtual battlespace. The connection between the embedded simulation system and the vehicle embedded computer is defined via an A/B kit interface, which is realized as a series of bi-directional data based TCP/IP messages. The embedded simulation and vehicle applications cannot be co-resident because they utilize different technologies and network communication protocols (with the exception of well-defined A/B kit interface).

The ultimate goal of this analysis is to provide a robust, extensible, standards-based embedded simulation capability that minimally impacts the operational characteristics of the vehicle and is consistent with the rest of the vetronics architecture. To meet that goal, the simulation infrastructure must support real-time quality of service, must be interoperable across languages and implementations, and must be available on many different platforms. Further, to minimize the impact on the rest of the system, the simulation infrastructure must be able to seamlessly share data with operational vehicle applications. This is contrary to the way developers currently design systems where they typically integrate three distinct architectures—vetronics, C4ISR, and simulation. Each of these usually uses different middleware products for distributed communication. Vetronics applications use real-time operating environments, C4ISR applications use CORBA or a similar commercial middleware product, and embedded simulations use the A-kit/B-kit approach described in Section 1. One way to harmonize these three architectures is to use real-time CORBA, in conjunction with some of the related emerging specifications (real-time notification and data distribution), as the distributed communications middleware for all vetronics, C4ISR, and embedded simulation applications

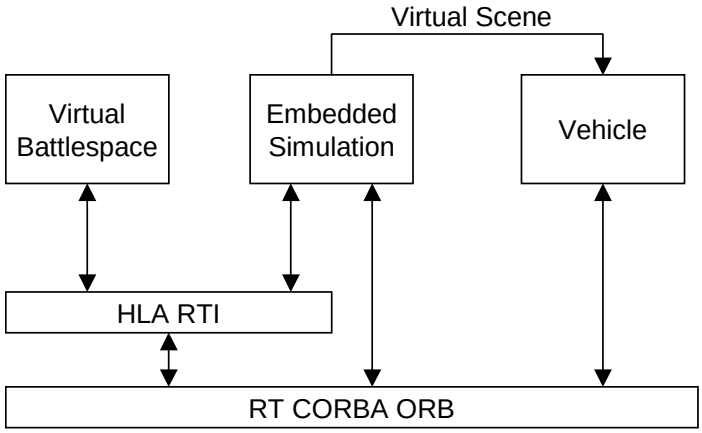

Figure 6 Proposed Integrated Architecture

As shown in Figure 6, the proposed architecture uses real-time CORBA as the common infrastructure bridging the embedded simulation and vehicle applications. Vehicle applications use real-time CORBA for distributed communication, as does the HLA RTI. This would allow a more efficient and maintainable coupling between the operational vehicle software and the embedded simulation capabilities, as developers need only integrate and maintain one middleware product. For example, consider a vehicle with three major states as shown in Figure 7.

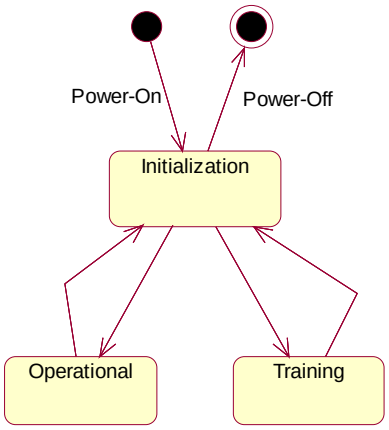

Figure 7 Example Vehicle States

In this example, transitions to the operational or training states can only occur from the initialization state. This makes it easy to setup the required resources for a single state at the same time. An integrated CORBA/HLA-based architecture could consist of application objects inheriting from a CORBA object as shown in Figure 8.

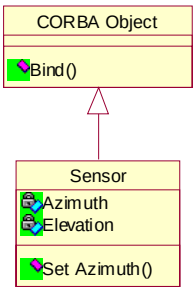

Figure 8 Diagram Example, Class Diagram

Then, depending on the state, i.e., training, or operational, a different implementation of the application object could be "bound" (resolved) at runtime to achieve either simulated or operational behavior as shown by the collaboration diagram in Figure 8. This approach provides a flexible, non-intrusive embedded simulation capability that is transparent to the vehicle applications. The actual vehicle software is literally unaware of whether or not the object being interacted with is "real" or simulated. The overall system is also more supportable and maintainable as both the embedded simulation capabilities and the remaining vehicle software share a common distribution infrastructure. Further, it is possible to specify the interface to the application objects in a way that hides the actual middleware (contains no visible references to CORBA), to preserve the investment in the application objects as both the middleware and simulation technologies evolve.

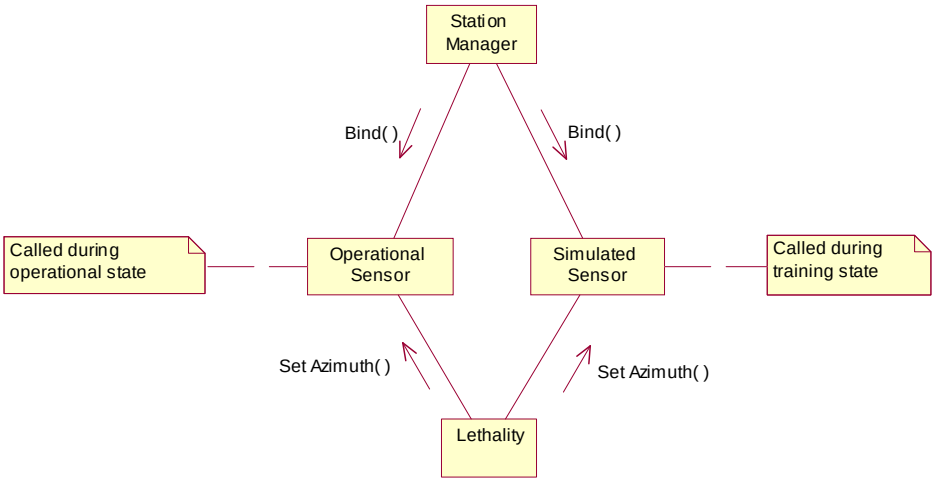

Figure 9 Example Collaboration

5 Summary and Conclusions

This paper presents an open standards based architecture that seamlessly integrates embedded simulation capabilities into a real-time embedded ground vehicle system. Of the two standards proposed, only CORBA is mature enough for use in ground combat vehicles. HLA, though promising, still has drawbacks in the areas of real-time computing, interoperability, and platform availability. The future addition of CORBA-based HLA run-time infrastructures, however, can alleviate those shortcomings. Further work is needed, however, to validate the architecture approach and ensure that both of these technologies can live in harmony within a single integrated platform. This validation can be accomplished in multiple stages with the first stage consisting of CORBA-based vehicle software and a non-CORBA, workstation-based RTI implementation. This would both ensure that the architecture approach is sound and would start to mature the application object interfaces. The full validation could occur as CORBA-based RTI implementations become available on multiple platforms.

6 References

[1] U.S. Department of Defense, *High Level Architecture Interface Specification*, Version 1.3, Draft 11, 20 April 1998.

[2] Object Management Group, *The Common Object Request Broker: Architecture and Specification*, Revision 2.5, Sept. 2001.

[3] Object Management Group, *Distributed Simulation Systems Specification*, Version 2.0, Nov. 2002.

[4] Thom McClean, Richard Fujimoto, and Brad Fitzgibbons, “Middleware for Real-Time Distributed Simulations,” *Concurrency and Computation*, Wiley Interscience, Submitted.

[5] Hui Zhao and Nicolas Georganas, "HLA Real-Time Extension," *Fifth IEEE International Workshop on Distributed Simulation and Real-Time Applications*, August 13 - 15, 2001. Cincinnati, Ohio.

[6] Arnold Buss and Leroy Jackson, "Distributed Simulation Modeling: A Comparison of HLA, CORBA, and RMI," *Proceedings of the 1998 Winter Simulation Conference*, Washington, DC.

[7] http://www.corba.org/industries/aerodef/aerodef.html.

[8] http://www.corba.org/industries/gov/gov.html

[9] Ed Schrum, "Use of RT CORBA by the US Army," *ISORC 2001*, May 2, 2001

[10] Michael K. J. Milligan, "Implementing COTS Open Systems Technology on AWACS," *Crosstalk: The Journal of Defense Software Engineering*, Sept. 2000.

[11] The Boeing Corporation, *Real-Time CORBA Trade Study*, D204-31159-1, 31 Jan. 2000.